

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES  
PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum  
Internationales Büro



(43) Internationales Veröffentlichungsdatum  
18. August 2005 (18.08.2005)

PCT

(10) Internationale Veröffentlichungsnummer  
**WO 2005/076129 A1**

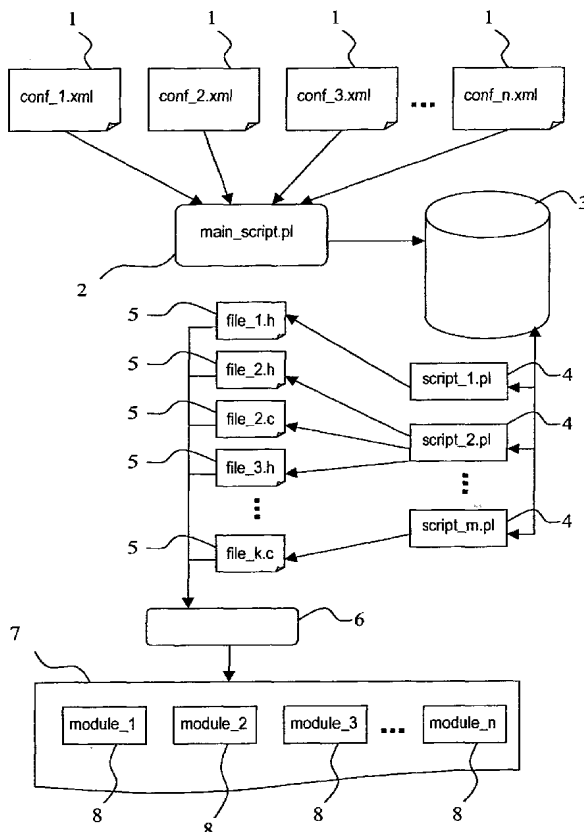
(51) Internationale Patentklassifikation<sup>7</sup>: **G06F 9/44**  
(21) Internationales Aktenzeichen: PCT/EP2005/050289  
(22) Internationales Anmeldedatum:  
24. Januar 2005 (24.01.2005)  
(25) Einreichungssprache: Deutsch  
(26) Veröffentlichungssprache: Deutsch  
(30) Angaben zur Priorität:  
10 2004 005 730.3 5. Februar 2004 (05.02.2004) DE  
(71) Anmelder (für alle Bestimmungsstaaten mit Ausnahme von  
US): **ROBERT BOSCH GMBH** [DE/DE]; Postfach 30 02  
20, 70442 Stuttgart (DE).

(72) Erfinder; und  
(75) Erfinder/Anmelder (nur für US): **SCHENK, Rene**  
[DE/DE]; Eichenweg 7, 71732 Tamm (DE). **BEUTER,**  
**Bjoern** [DE/DE]; Horber Str. 35, 71149 Bondorf (DE).  
**SCHNEIDER, Klaus** [DE/DE]; Hanseatenstr. 22, 71640  
Ludwigsburg (DE). **ILLG, Bernd** [DE/DE]; Geranienstr.  
7/1, 75031 Eppingen (DE).  
(81) Bestimmungsstaaten (soweit nicht anders angegeben, für  
jede verfügbare nationale Schutzrechtsart): AE, AG, AL,  
AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH,  
CN, CO, CR, CU, CZ, DK, DM, DZ, EC, EE, EG, ES, FI,  
GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE,  
KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD,  
MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG,

[Fortsetzung auf der nächsten Seite]

(54) Title: METHOD FOR CONFIGURING A COMPUTER PROGRAM

(54) Bezeichnung: VERFAHREN ZUR KONFIGURATION EINES COMPUTERPROGRAMMS



(57) Abstract: The aim of the invention is to easily and flexibly configure a computer program comprising at least one functional unit. To this end, the invention provides a method having the following steps: creating at least one implementation-independent configuration file and/or altering information stored in the at least one implementation-dependent configuration file; automatically establishing and/or automatically updating configuration data, which are stored in a configuration data container, based on the information stored in the at least one implementation-independent configuration file; automatically creating at least one implementation-dependent configuration file based on the configuration data stored in the configuration data container, and; automatically configuring the at least one functional unit based on the information stored in the at least one implementation-dependent configuration file.

(57) Zusammenfassung: Um ein mindestens eine Funktionseinheit umfassendes Computerprogramm besonders einfach und flexibel zu konfigurieren wird ein Verfahren vorgeschlagen, das die folgenden Schritte umfasst: Erstellen mindestens einer implementierungsunabhängigen Konfigurationsdatei und/oder Änderung von in der mindestens einen implementierungsunabhängigen Konfigurationsdatei abgelegten Informationen; Automatischer Aufbau und/oder automatische Aktualisierung von in einem Konfigurationsdatencontainer abgelegten Konfigurationsdaten in Abhängigkeit von den in der mindestens einen implementierungsunabhängigen Konfigurationsdatei abgelegten Informationen; Automatisches Erzeugen mindestens einer implementierungsabhängigen Konfigurationsdatei in Abhängigkeit von den in dem Konfigurationsdatencontainer abgespeicherten Konfigurationsdaten; Automatische Konfiguration der mindestens einen Funktionseinheit in Abhängigkeit von in der mindestens einen

implementierungsabhängigen Konfigurationsdatei abgelegten Informationen.

WO 2005/076129 A1



PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**(84) Bestimmungsstaaten** (soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL,

**Veröffentlicht:**

— mit internationalem Recherchenbericht

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

5

10 Verfahren zur Konfiguration eines Computerprogramms

Die Erfindung betrifft ein Verfahren zur Konfiguration eines mindestens eine Funktionseinheit umfassenden Computerprogramms.

15

Die Erfindung betrifft auch ein Softwaresystem zur Konfiguration eines mindestens eine Funktionseinheit umfassenden Computerprogramms.

20 Stand der Technik

Moderne Computerprogramme werden überwiegend so programmiert, dass sie in einem möglichst <sup>(14)</sup> breiten Anwendungsbereich einsetzbar sind. Der Anwendungsbereich wird einerseits durch die zur Verfügung gestellten Funktionalitäten, die wiederum möglichst viele Benutzerwünsche abdecken sollen, und andererseits durch die zugrunde liegende Hardware, auf der das Computerprogramm ablaufen soll, bestimmt. Die zugrunde liegende Hardware bezeichnet hierbei unterschiedliche Computersysteme, die in unterschiedlichen Bereichen eingesetzt werden, aus unterschiedlichen Komponenten (beispielsweise Prozessoren oder Bussystemen) aufgebaut sind und/oder über unterschiedliche Peripheriegeräte verfügen.

35

Unterschiedliche Funktionalitäten können sich aus unterschiedlichen Gegebenheiten der zugrunde liegenden

Hardware ergeben oder aus unterschiedlichen Benutzerwünschen. Eine Anpassung und damit eine Spezialisierung eines Computerprogramms an eine zugrunde liegende Hardware und an bestimmte Benutzerwünsche umfasst  
5 eine sogenannte Konfiguration des Computerprogramms.

Eine Konfiguration umfasst beispielsweise das Aktivieren oder Deaktivieren einzelner Funktionen des Computerprogramms, das Setzen von Startwerten für bestimmte  
10 Variablen oder das Vorgeben und Spezifizieren bestimmter Variablentypen.

Es ist bekannt, die in einem Computerprogramm verwendeten Variablen und Funktionen in einer sogenannten Header-Datei  
15 zu deklarieren und eine Konfiguration des Computerprogramms durchzuführen, indem einzelne Variablen oder Funktionsbezeichner in der Header-Datei verändert werden. Beispielsweise ist es möglich, einem in dem Computerprogramm verwendeten und in der Header-Datei  
20 deklarierten Funktionsbezeichner <sup>in</sup> in Abhängigkeit von einer bestimmten Konfiguration eine spezielle Funktion zuzuweisen.

Üblicherweise werden Computerprogramme in einer sogenannten  
25 Hochsprache, beispielsweise C, C++, Scheme oder JAVA, erstellt. Üblicherweise wird ein in einer Hochsprache erstelltes Computerprogramm als Source-Code bezeichnet. Um ein derartiges Computerprogramm auf einem Computer ausführen zu können, muss aus dem Source-Code ein  
30 sogenannter Maschinencode erzeugt werden, der Anweisungen enthält, die von dem Prozessor des Computers ausführbar sind. Maschinencode kann durch sogenanntes Interpretieren oder Kompilieren des Source-Codes erzeugt werden.

35 Typischerweise umfasst ein Computerprogramm eine Vielzahl

von Funktionseinheiten. Der Source-Code einer oder mehrerer Funktionseinheiten ist dabei in einer Datei abgespeichert. Einer oder mehrerer derartiger Dateien ist eine Header-Datei zugeordnet. Somit besteht ein Computerprogramm

5 typischerweise aus einer Vielzahl von Dateien. Eine Konfiguration eines derartigen Computerprogramms, die durch Änderungen innerhalb einzelner Header-Dateien durchgeführt wird, ist deshalb sehr unübersichtlich und kann häufig nur von dem Ersteller des Source-Codes durchgeführt werden.

10 Zudem muss eine Dokumentation aller Header-Dateien erstellt werden, was sehr aufwendig ist, wobei selbst die Dokumentation meist sehr unübersichtlich ist.

Es ist auch bekannt, zur Konfiguration eines

15 Computerprogramms diesem eine spezielle Funktionseinheit zuzuordnen, mittels der eine Konfiguration des gesamten Computerprogramms beispielsweise durch Änderung der Werte von vorgegebenen Parametern möglich ist. Die Funktionseinheit kann beispielsweise aus dem ablaufenden

20 Computerprogramm heraus aufgerufen und zur Konfiguration des Computerprogramms ausgeführt werden. Eine derartige für die Konfiguration des Computerprogramms vorgesehene Funktionseinheit erlaubt jedoch nur eine Konfiguration innerhalb vorgegebener Bereichsgrenzen. Eine Konfiguration

25 des Computerprogramms beispielsweise zur Anpassung des Computerprogramms auf eine neue Hardware oder zur Anpassung des Computerprogramms an neue Benutzerwünsche ist mit einer derartigen Funktionseinheit nicht möglich. Ferner muss die zur Konfiguration verwendete Funktionseinheit speziell für

30 das betreffende Computerprogramm entwickelt werden und kann nicht für andere Computerprogramme verwendet werden.

Aufgabe der Erfindung ist es daher, eine Möglichkeit zu schaffen, die eine möglichst übersichtliche und flexible

35 Konfiguration eines Computerprogramms erlaubt.

Die Aufgabe wird durch ein Verfahren der eingangs genannten Art gelöst, das die folgenden Schritte umfasst:

- 5       - Erstellen mindestens einer  
          implementierungsunabhängigen Konfigurationsdatei  
          und/oder Änderung von in der mindestens einen  
          implementierungsunabhängigen Konfigurationsdatei  
          abgelegten Informationen;
- 10       - automatischer Aufbau und/oder automatische  
          Aktualisierung von in einem  
          Konfigurationsdatencontainer abgelegten  
          Konfigurationsdaten in Abhängigkeit von den in der  
          mindestens einen implementierungsunabhängigen  
15       Konfigurationsdatei abgelegten Informationen;
- automatisches Erzeugen mindestens einer  
          implementierungsabhängigen Konfigurationsdatei in  
          Abhängigkeit von den in dem  
          Konfigurationsdatencontainer abgespeicherten  
20       Konfigurationsdaten;
- automatische Konfiguration der mindestens einen  
          Funktionseinheit in Abhängigkeit von in der mindestens  
          einen implementierungsabhängigen Konfigurationsdatei  
          abgelegten Informationen.

25

#### Vorteile der Erfindung

- Die eine Konfiguration bestimmenden Daten werden also  
unabhängig von einer beabsichtigten konkreten
- 30   Implementierung in einer oder mehreren  
      implementierungsunabhängigen Konfigurationsdateien  
      abgelegt. Die Implementierungsunabhängigkeit dieser  
      Konfigurationsdatei ermöglicht insbesondere eine abstrakte  
      Beschreibung der abgelegten Informationen. Dies ermöglicht
- 35   es, die für die Konfiguration des Computerprogramms

relevanten Informationen besonders gut lesbar abzulegen und damit die Konfiguration deutlich zu vereinfachen. Dadurch, dass diese Konfigurationsdatei implementierungsunabhängig ist, ist es insbesondere möglich, auf einfache Weise eine Konfiguration des Computerprogramms zu realisieren, so dass das Computerprogramm beispielsweise auf einem neuen Computersystem ablauffähig ist, dessen genaue Parameter bei der Erstellung des Computerprogramms noch gar nicht bekannt waren.

10

Der Konfigurationsdatencontainer ermöglicht es, alle für eine Konfiguration relevanten Daten zentral vorzuhalten. Mittels der in dem Konfigurationsdatencontainer abgelegten Konfigurationsdaten wird automatisch mindestens eine implementierungsabhängige Konfigurationsdatei erzeugt. In der implementierungsabhängigen Konfigurationsdatei findet bezüglich der implementierungsunabhängigen Konfigurationsdatei eine Konkretisierung einzelner oder mehrere Parameterwerte statt. Bei einer derartigen Konkretisierung werden beispielsweise relative Werte durch absolute Werte ersetzt. Ebenso können einzelnen Werten oder Datenbereichen bestimmte Datentypen oder -strukturen zugeordnet werden. Die implementierungsabhängige Konfigurationsdatei berücksichtigt folglich implementierungsabhängige Eigenschaften, wie beispielsweise eine oder mehrere bei der Programmierung des Source-Codes verwendeten Programmiersprachen oder Eigenschaften der Hardware, auf der das Computerprogramm ablaufen soll.

30

Der Aufbau beziehungsweise die Aktualisierung des Konfigurationsdatencontainers mittels der in den implementierungsunabhängigen Konfigurationsdateien abgelegten Informationen kann beispielsweise mittels sogenannter Scripte durchgeführt werden. Dabei bezeichnet ein Script eine Sequenz von Befehlen, die von einem

35

speziellen Computerprogramm ausgeführt werden können.  
Derartige spezielle Computerprogramme sind beispielsweise  
AWK oder Perl. Diese speziellen Computerprogramme können  
auch eingesetzt werden, um aus den in dem  
5 Konfigurationsdatencontainer abgelegten Konfigurationsdaten  
implementierungsabhängige Konfigurationsdateien zu  
erzeugen.

Ein wesentlicher Bestandteil der Erfindung ist also die  
10 Erkenntnis, dass die Konfiguration eines Computerprogramms  
entscheidend verbessert werden kann, indem zwischen einen  
Benutzer (Konfigurator) und dem Computerprogramm eine  
abstrakte Beschreibung der auszuführenden Konfiguration in  
der implementierungsunabhängigen Konfigurationsdatei  
15 vorgesehen ist, die der Konfiguration zugrunde gelegt wird.  
Anhand der implementierungsunabhängigen Konfigurationsdatei  
wird automatisch eine implementierungsabhängige  
Konfigurationsdatei erstellt, die dann zur Konfiguration  
des Computerprogramms herangezogen wird. Das  
20 erfindungsgemäße Verfahren ermöglicht es also, die eine  
Konfiguration beschreibenden Informationen abstrakt und  
damit besonders gut lesbar anzugeben. Durch die  
Unabhängigkeit jeglicher Implementierungsdetails wird  
ferner eine besonders hohe Flexibilität erreicht.

25 In einer vorteilhaften Weiterbildung des Verfahrens wird  
mindestens eine Abhängigkeitsinformation, die eine  
Abhängigkeit von mindestens zwei in dem  
Konfigurationsdatencontainer vorliegenden  
30 Konfigurationsdaten beschreibt, automatisch erzeugt. Die  
mindestens eine implementierungsabhängige  
Konfigurationsdatei wird in Abhängigkeit der mindestens  
einen Abhängigkeitsinformation erzeugt.



Abhängigkeitsinformationen können beispielsweise beschreiben, ob sich die Änderung eines Konfigurationsparameters auf einen anderen Konfigurationsparameter auswirkt. Wird zum Beispiel für  
5 eine Funktionseinheit eine Resource ausschließlich reserviert, so steht sie während der Ausführung der Funktionseinheit anderen Funktionseinheiten nicht zur Verfügung. Mit Abhängigkeitsinformationen kann ermittelt werden, welche Funktionseinheiten eine bestimmte Resource  
10 benötigen und damit nicht gleichzeitig ablaufen können. Abhängigkeitsinformationen können folglich auch zur Ressourcenverwaltung verwendet werden.

In einer bevorzugten Ausführungsform des Verfahrens werden  
15 mehrere implementierungsunabhängige Konfigurationsdateien erstellt und jede der implementierungsunabhängigen Konfigurationsdateien wird mindesten einer Funktionseinheit zugeordnet. Dies ermöglicht eine besonders einfache Konfiguration dadurch, dass die als Informationen in den  
20 implementierungsunabhängigen Konfigurationsdateien abgelegten Konfigurationsparameter besonders einfach aufgefunden und verändert werden können. Beispielsweise ist es möglich, die eine Konfiguration bestimmenden Informationen, also die Konfigurationsparameter, nach der  
25 von diesen beeinflussten Funktionalität oder Hardware zu ordnen. Ferner wird dadurch eine besonders einfache Anpassung der implementierungsunabhängigen Konfigurationsdateien an neu hinzugekommene Funktionseinheiten ermöglicht. Im einfachsten Fall wird  
30 einer neu hinzugekommenen Funktionseinheit eine spezielle implementierungsunabhängige Konfigurationsdatei zugeordnet.

Vorteilhafterweise werden mehrere implementierungsabhängige Konfigurationsdateien erzeugt und es wird jede der  
35 implementierungsabhängigen Konfigurationsdatei mindestens

einer Funktionseinheit zugeordnet. Eine derartige Strukturierung der implementierungsabhängigen Konfigurationsdateien erhöht die Übersichtlichkeit der erzeugten implementierungsabhängigen Konfigurationsdateien.

5 Ist der Source-Code derartig strukturiert, dass sich eine oder mehrere Funktionseinheiten in unterschiedlichen Dateien befinden, so kann erreicht werden, dass jeder der Dateien des Source-Codes eine implementierungsabhängige Konfigurationsdatei zugeordnet ist. Eine besonders  
10 übersichtliche Strukturierung kann auch dadurch erreicht werden, dass jeder implementierungsunabhängigen Konfigurationsdatei je eine implementierungsabhängige Konfigurationsdatei zugeordnet ist.

15 Vorzugsweise wird die mindestens eine implementierungsabhängige Konfigurationsdatei in Abhängigkeit von mindestens einer Eigenschaft einer Hardware, auf der eine Installation zumindest eines Teils des konfigurierten Computerprogramms ermöglicht werden  
20 soll, erzeugt. Eine derartige Hardwareeigenschaft kann beispielsweise die Anzahl von zur Verfügung stehenden Prozessoren oder Art und Anzahl von an die Hardware angeschlossenen Sensoren sein. Werden derartige Hardwareeigenschaften bei der Erzeugung der  
25 implementierungsabhängigen Konfigurationsdateien berücksichtigt, so kann eine besonders präzise Konfiguration des Computerprogramms erfolgen. Damit ist es beispielsweise möglich, insbesondere mit  
30 Abhängigkeitsinformationen eine hinsichtlich der Ausführungsgeschwindigkeit optimierte Konfiguration automatisch zu erstellen.

In einer bevorzugten Ausführungsform wird die mindestens eine implementierungsabhängige Konfigurationsdatei in  
35 Abhängigkeit von dem Ergebnis einer Plausibilitätsprüfung

erzeugt. Eine Plausibilitätsprüfung kann beispielsweise die Prüfung umfassen, ob eine von einer Funktionseinheit benötigte Resource überhaupt zur Verfügung steht.

- 5 Vorzugsweise wird zur Durchführung der Plausibilitätsprüfung die mindestens eine Hardwareeigenschaft verwendet. Dadurch kann der Automatisierungsgrad besonders gut erhöht werden und eine zuverlässige Konfiguration des Computerprogramms erreicht  
10 werden. Sieht eine Funktionseinheit beispielsweise eine Erfassung von Messwerten vor, so kann überprüft werden, ob geeignete Sensoren vorhanden sind und ob diese die angeforderte Messgenauigkeit zur Verfügung stellen. In diesem Fall ist es beispielsweise vorstellbar, dass  
15 automatisch eine Konfiguration des Sensors durchgeführt wird.

In einer weiteren bevorzugten Ausführungsform wird automatisch eine Dokumentation erstellt. Die Dokumentation  
20 beschreibt die innerhalb der mindestens einen implementierungsunabhängigen Konfigurationsdatei und/oder der mindestens einen implementierungsabhängigen Konfigurationsdatei abgelegten Informationen. Derartig automatisch erzeugte Dokumentationen erhöhen die  
25 Wartbarkeit des Computerprogramms einerseits und ermöglichen es andererseits besonders einfach, eine durchgeführte Konfiguration nachzuvollziehen. Durch die automatische Erzeugung der Dokumentation ist sichergestellt, dass diese mit der tatsächlichen  
30 Konfiguration übereinstimmt. Soll eine neue Konfiguration des Computerprogramms durchgeführt werden, so kann anhand einer derartigen Dokumentation besonders einfach festgestellt werden, welche Parameterwerte geändert werden müssen.

Vorzugsweise wird die mindestens eine implementierungsunabhängige Konfigurationsdatei in einem XML-basierten Format erstellt. XML (Extensible Markup Language) ist eine standardisierte Metasprache, die es ermöglicht, strukturierte Sprachen zu erzeugen. Ist die mindestens eine implementierungsunabhängige Konfigurationsdatei in einer XML-konformen, strukturierten Sprache erstellt, so wird eine Konfiguration dadurch erleichtert, dass eine derartige implementierungsunabhängige Konfigurationdatei besonders gut lesbar ist. Ferner ist eine derartige Konfigurationsdatei auch besonders gut maschinenlesbar. Insbesondere existiert eine Vielzahl von teilweise ebenfalls standardisierten Software-Werkzeugen (Tools), mittels derer eine Bearbeitung und Verarbeitung von in einem XML-basierten Format erstellten Dateien möglich ist.

In einer bevorzugten Ausführungsform des Verfahrens wird in Abhängigkeit von den Konfigurationsdaten automatisch ermittelt, ob eine von dem Computerprogramm umfasste Funktionseinheit von dem Computerprogramm benötigt wird und eine Konfiguration dieser Funktionseinheit nur durchgeführt, falls die Funktionseinheit von dem Computerprogramm benötigt wird. Dies ermöglicht es, eine Konfiguration besonders schnell durchzuführen dadurch, dass nur derartige Funktionseinheiten tatsächlich konfiguriert werden, die bei einer Ausführung des konfigurierten Computerprogramms tatsächlich benötigt werden. Des Weiteren wird dadurch erreicht, dass das konfigurierte Computerprogramm möglichst wenig Speicherplatz in Anspruch nimmt, da beispielsweise eine Übersetzung von Source-Code in Maschinencode nur für derartige Funktionseinheiten veranlasst wird, die tatsächlich angewendet werden sollen.

Die Aufgabe wird auch durch ein Softwaresystem der eingangs genannten Art gelöst. Dabei weist das Softwaresystem auf:

- 5       - mindestens eine implementierungsunabhängige Konfigurationsdatei;
- einen Konfigurationsdaten umfassenden Konfigurationsdatencontainer und/oder Mittel zum Erstellen eines Konfigurationsdatencontainers in Abhängigkeit von in der mindestens einen
- 10       implementierungsunabhängigen Konfigurationsdatei abgelegten Informationen;
- Mittel zum Ändern und/oder Auslesen von Konfigurationsdaten aus dem Konfigurationsdatencontainer;
- 15       - Mittel zum automatischen Erzeugen mindestens einer implementierungsabhängigen Konfigurationsdatei in Abhängigkeit von in dem Konfigurationsdatencontainer abgespeicherten Konfigurationsdaten; und
- Mittel zum automatischen Konfigurieren der mindestens
- 20       einen Funktionseinheit in Abhängigkeit von in der implementierungsabhängigen Konfigurationsdatei abgelegten Informationen.

Vorzugsweise weist das Softwaresystem Mittel zur

25       Durchführung des erfindungsgemäßen Verfahrens auf.

Die Realisierung der Erfindung in Form eines Softwaresystems ist hierbei von besonderer Bedeutung. Dabei ist das Softwaresystem auf einem Rechenggerät, insbesondere

30       auf einem Mikroprozessor ablauffähig und zur Ausführung des erfindungsgemäßen Verfahrens geeignet. In diesem Fall wird also die Erfindung durch das Softwaresystem realisiert, so dass das Softwaresystem in gleicher Weise die Erfindung darstellt wie das Verfahren, zu dessen Ausführung das

35       Softwaresystem geeignet ist. Das Softwaresystem ist

vorzugsweise auf einem Speicherelement abgespeichert. Das Speicherelement kann als Random-Access-Memory, Read-Only-Memory oder Flash-Memory ausgebildet sein. Das Speicherelement kann auch als Digital Versatile Disc (DVD),  
5 Compact Disc (CD) oder als Festplatte (Hard Disc) ausgebildet sein.

#### Zeichnungen

10 Weitere Merkmale, Anwendungsmöglichkeiten und Vorteile der Erfindung ergeben sich aus der nachfolgenden Beschreibung von Ausführungsbeispielen der Erfindung, die in der  
Zeichnung dargestellt sind. Dabei bilden alle beschriebenen oder dargestellten Merkmale für sich oder in beliebiger  
15 Kombination den Gegenstand der Erfindung, unabhängig von ihrer Zusammenfassung in den Patentansprüchen oder deren Rückbeziehung sowie unabhängig von ihrer Formulierung beziehungsweise Darstellung in der Beschreibung beziehungsweise in der Zeichnung. Es zeigen:

- 20 ..
- Figur 1 eine Ausführungsform eines Softwaresystems zur Durchführung des erfindungsgemäßen Verfahrens;  
und
- 25 Figur 2 ein schematisiertes Ablaufdiagramm einer Ausführungsform des erfindungsgemäßen Verfahrens.

#### Beschreibung der Ausführungsbeispiele

30 In Figur 1 ist ein Softwaresystem zur Durchführung des erfindungsgemäßen Verfahrens dargestellt. Das Softwaresystem weist mehrere implementierungsunabhängige Konfigurationsdateien 1 auf. Jeder Konfigurationsdatei 1 ist ein Dateiname zugeordnet. Die in Figur 1 dargestellten  
35 implementierungsunabhängigen Konfigurationsdateien tragen

beispielsweise die Dateinamen conf\_1.xml, conf\_2.xml, conf\_3.xml bis conf\_n.xml. Die Dateiendung .xml weist darauf hin, dass die implementierungsunabhängigen Konfigurationsdateien 1 in einem XML-basierten Format vorliegen. Eine in einem XML-basierten Format vorliegende Text-Datei ermöglicht es, die Text-Datei nach vorgebbaren Regeln zu strukturieren. Eine derartig strukturierte Text-Datei kann besonders gut manuell und maschinell gelesen und verarbeitet werden.

10

Die implementierungsunabhängigen Konfigurationsdateien 1 werden einem Script 2 zugeführt. Das Script 2 ist beispielsweise als sogenanntes Perl-Script ausgebildet. Perl ist eine Interpretersprache, deren Syntax auf der Programmiersprache C basiert und die von dem jeweiligen Betriebssystem zur Verfügung gestellte Dienstprogramme verwendet.

15

Mittels des Scripts 2 werden die implementierungsunabhängigen Konfigurationsdateien 1 gelesen und die darin abgelegten Informationen extrahiert und in einem Konfigurationsdatencontainer 3 abgelegt. Gleichzeitig werden auch eventuell vorhandene Abhängigkeiten zu den weiteren Konfigurationsscripten 4 bestimmt und abgelegt.

20

25

Mit dem Bezugszeichen 4 sind weitere Konfigurationsscripte dargestellt. Diese sind ebenfalls als Pearl-Scripte ausgebildet. Es ist ebenso vorstellbar, dass eines oder mehrere der weiteren Konfigurationsscripte 4 ein ausführbares Computerprogramm (Maschinencode) ist oder in einer anderen Scriptsprache, beispielsweise AWK, vorliegt.

30

Mit dem Bezugszeichen 5 sind implementierungsabhängige Konfigurationsdateien bezeichnet. Die

35

implementierungsabhängigen Konfigurationsdateien 5 sind beispielsweise in der Programmiersprache codiert, in der auch der zu konfigurierende Source-Code programmiert ist. Derartige implementierungsabhängige Konfigurationsdateien 5 können von einem Compiler 6 verarbeitet werden.

Mit dem Bezugszeichen 7 ist ein Computerprogramm dargestellt, das mehrere Funktionseinheiten 8 aufweist.

10 Die Funktionsweise des erfindungsgemäßen Softwaresystems wird anhand des in Figur 2 dargestellten Ablaufdiagramms beschrieben.

Das in Figur 2 dargestellte Ablaufdiagramm eines  
15 erfindungsgemäßen Verfahrens zur Konfiguration eines Computerprogramms startet in einem Schritt 100. In einem Schritt 101 werden implementierungsunabhängige Konfigurationsdateien 1 erstellt beziehungsweise geändert. Die implementierungsunabhängigen Konfigurationsdateien 1  
20 zeichnen sich insbesondere dadurch aus, dass es möglich ist, mittels der dort abgelegten Informationen konkrete Konfigurationswerte beziehungsweise Konfigurationsparameter abstrakt zu beschreiben. Konkrete Konfigurationswerte können beispielsweise den Messbereich eines Sensormoduls  
25 zur Messung einer elektrischen Spannung bestimmen. Beispielsweise ist es möglich, einen Messbereich abstrakt mit den Werten 3 - 5 Volt anzugeben. Die daraus zu erzeugenden, implementierungsabhängigen Werte des Messbereichs so, wie es die zu konfigurierende  
30 Funktionseinheit 8 erwartet, können jedoch beispielsweise zwischen 10.000 und 20.000 liegen. Eine das Sensormodul steuernde Funktionseinheit 8 des Computerprogramm müsste in diesem Fall beispielsweise mittels der konkreten Konfigurationswerte 10.000 und 20.000 konfiguriert werden ,



um eine Messung in einem Messbereich von 3 - 5 Volt zu ermöglichen.

Die in dem Schritt 101 erstellten beziehungsweise  
5 geänderten implementierungsunabhängigen  
Konfigurationsdateien 1 sind beispielsweise in einem XML-  
basierten Format erstellt. Ein derartiges Format ermöglicht  
es besonders gut, eine übersichtliche Strukturierung der  
implementierungsunabhängigen Konfigurationsdateien 1 zu  
10 erreichen. Dies erhöht die Lesbarkeit der  
implementierungsunabhängigen Konfigurationsdateien 1 und  
vereinfacht die Änderung der implementierungsunabhängigen  
Konfigurationsdateien 1 beispielsweise dadurch dass zu  
ändernde Konfigurationsdaten schnell aufgefunden werden  
15 können. Es ist möglich, auch für ein besonders großes  
Computerprogramm, für dessen Konfiguration eine Vielzahl  
von Konfigurationsdaten notwendig sind, nur eine einzige  
implementierungsunabhängige Konfigurationsdatei vorzusehen.  
Eine Strukturierung der in der implementierungsunabhängigen  
20 Konfigurationsdatei 1 abgelegten Informationen kann dabei  
durch geeignete XML-Strukturen erreicht werden. Besonders  
vorteilhaft jedoch ist es, mehrere  
implementierungsunabhängige Konfigurationsdateien  
vorzusehen. Jede dieser implementierungsunabhängigen  
25 Konfigurationsdateien 1 kann beispielsweise einer oder  
mehrerer Funktionseinheiten 8 zugeordnet sein. Dadurch ist  
es möglich, das Erstellen beziehungsweise Ändern der  
implementierungsunabhängigen Konfigurationsdateien  
besonders übersichtlich durchzuführen. Des Weiteren wird  
30 dadurch eine Wiederverwendbarkeit einzelner  
implementierungsunabhängiger Konfigurationsdateien erhöht.  
Dies ist insbesondere für Projekte vorteilhaft, bei denen  
auch einzelne Funktionseinheiten 8 des Source-Codes  
wiederverwendet werden sollen.

In einem Schritt 102 wird der Konfigurationsdatencontainer 3 aufgebaut, beziehungsweise aktualisiert. Dies geschieht durch Abarbeiten der in dem Script 2 aufgelisteten Anweisungen. Das Script 2 veranlasst zunächst, dass die unabhängigen Konfigurationsdateien 1 eingelesen werden. Liegt den implementierungsunabhängigen Konfigurationsdateien 1 ein strukturiertes Format, beispielsweise ein XML-basiertes Format zugrunde, so kann mittels des Scripts 2 besonders gut eine syntaktische und/oder semantische Analyse des Inhalts der implementierungsunabhängigen Konfigurationsdateien 1 durchgeführt werden. Damit können beispielsweise Fehler bei der Angabe von Konfigurationsdaten erkannt werden. Vorzugsweise weist das XML-basierte Format der implementierungsunabhängigen Konfigurationsdateien 1 eine hierarchische Struktur auf, die sich vorteilhafterweise an der Struktur der Funktionseinheiten 8 selbst, deren Abhängigkeiten und/oder deren thematische Nähe orientiert. Mittels des Scripts 2 können Fehler beim Aufbau dieser hierarchischen Struktur und damit auch beim Aufbau des Source-Codes selbst erkannt werden.

Vorteilhafterweise weist der Schritt 102 eine Behandlung der aufgefundenen Fehler auf. Dies kann beispielsweise durch die Ausgabe von Fehlerinformationen geschehen. Es ist ebenso vorstellbar, dass stochastische Verfahren zur Beseitigung von Fehlern eingesetzt werden.

Das Script 2 extrahiert in dem Schritt 102 die in den implementierungsunabhängigen Konfigurationsdateien 1 vorhandenen Konfigurationsdaten und speichert diese in dem Konfigurationsdatencontainer 3 ab. Der Konfigurationsdatencontainer 3 kann dabei beispielsweise als Datenbank ausgebildet sein. Es ist ebenso vorstellbar, dass der Konfigurationsdatencontainer 3 als in einem

Speicherbereich vorgehaltene Datenstruktur innerhalb des erfindungsgemäßen Softwaresystems realisiert ist, wobei gewährleistet ist, dass das Script 2 schreibenden und lesenden Zugriff auf die in dem

5 Konfigurationsdatencontainer 3 abgelegten Konfigurationsdaten hat.

In einem Schritt 103 werden Abhängigkeiten ermittelt. Eine derartige Abhängigkeit kann beispielsweise beschreiben,  
10 welche Funktionseinheiten 8 des Computerprogramms bei der vorliegenden Konfiguration tatsächlich abgearbeitet werden müssen. Mittels dieser Abhängigkeiten kann entschieden werden, ob in einem der folgenden Schritte für eine bestimmte Funktionseinheit 8 überhaupt eine  
15 implementierungsabhängige Konfigurationsdatei erzeugt werden muss. Abhängigkeiten können ferner beschreiben, welche konkreten Konfigurationsdaten von welchen abstrakten Konfigurationsdaten abhängen. So ist es vorstellbar, dass die Änderung eines abstrakten Konfigurationsdatums in einer  
20 implementierungsunabhängigen Konfigurationsdatei eine Änderung einer Mehrzahl von konkreten Konfigurationsdaten bewirkt.

Abhängigkeiten können sich auch ergeben, wenn die weiteren  
25 Scripte 4 ihrerseits den Konfigurationscontainer 3 verändern. Somit muss die richtige Aufrufreihenfolge (Aktivierungssequenz) der Scripte 4 ermittelt und abgelegt werden. Abhängigkeiten können auch Beziehungen zwischen einer oder mehrerer Hardwarekomponenten und einzelner  
30 Konfigurationsdaten beschreiben. Dies ermöglicht es beispielsweise zu erkennen, ob eine vorgesehene Konfiguration auf einer bestimmten Hardware tatsächlich ablauffähig ist.

In einem Schritt 104 wird eine Plausibilitätsprüfung durchgeführt. Dabei wird insbesondere anhand der in Schritt 103 ermittelten Abhängigkeiten überprüft, ob die mittels der implementierungsunabhängigen Konfigurationsdateien 1 vorgegebene Konfiguration Fehler aufweist. Ist dies der Fall, so wird in den Schritt 101 zurückverzweigt, in welchem eine Änderung der implementierungsunabhängigen Konfigurationsdateien 1 mit dem Zwecke einer Fehlerbeseitigung durchgeführt wird. Werden in dem Schritt 104 keine Fehler erkannt, so wird zu einem Schritt 105 verzweigt.

In dem Schritt 105 werden die implementierungsabhängigen Konfigurationsdateien 5 erzeugt. Dazu werden mittels eines Scripts 4 oder einer Mehrzahl von Scripten 4 zunächst die in dem Konfigurationsdatencontainer 3 abgelegten Konfigurationsdaten abgerufen. In dem vorliegenden Ausführungsbeispiel sind die Scripte 4 als Pearl-Scripte ausgebildet. Mittels der Scripte 4 werden insbesondere in dem Konfigurationsdatencontainer 3 abgelegte abstrakte Konfigurationsdaten in konkrete Konfigurationsdaten umgewandelt, die dann in den implementierungsabhängigen Konfigurationsdateien 5 abgelegt werden. Vorzugsweise werden dabei auch die in dem Schritt 103 ermittelten Abhängigkeiten verwendet.

Die in dem Schritt 105 erzeugten implementierungsabhängigen Konfigurationsdateien 5 können beispielsweise Header-Dateien (file\_.h, file\_2.h, file\_3.h in Figur 1) sein. Ebenso können die erzeugten implementierungsabhängigen Konfigurationsdateien 5 auch Source-Code enthalten (file\_2.c, file\_k.c in Figur 1). Typischerweise sind die von den Scripten 4 aus den abstrakten Konfigurationsdaten erzeugten konkreten Konfigurationsdaten durch Wertzuweisungen für Variablen und/oder Funktionsparameter

sowie als Anweisungen in einer Programmiersprache realisiert. Dabei entspricht die Programmiersprache der Programmiersprache, in der die Funktionseinheiten 8 des Computerprogramms 7 codiert sind. Sind die

5 Funktionseinheiten 8 des Computerprogramm 7 beispielsweise in der Programmiersprache C++ codiert, so können die konkreten Konfigurationsdaten beispielsweise durch sogenannte #Define-Anweisungen oder durch die Definition konstanter Variablen realisiert werden. Mittels der Scripte

10 4 ist es auch möglich, in Abhängigkeit der in dem Konfigurationsdatencontainer 3 abgespeicherten Konfigurationsdaten Funktionen zu erzeugen, die komplexe Aufgaben - wie beispielsweise die Initialisierung von Hardwarekomponenten oder die Prüfung auf das Vorhandensein

15 einzelner Softwarekomponenten oder Hardwarekomponenten - übernehmen und selbst als Source-Code in einer höheren Programmiersprache realisiert sind. Dieser Source-Code kann dann in einer oder mehreren implementierungsabhängigen Konfigurationsdateien (file\_2.c, file\_k.c in Figur 1)

20 abgelegt werden. Dazu kann ein Script 4 beispielsweise ein sogenanntes Template enthalten, das beispielsweise aus Anweisungen in C++ besteht, die in Abhängigkeit der in dem Konfigurationsdatencontainer 3 abgespeicherten Konfigurationsdaten aktualisiert werden und in einer

25 implementierungsabhängigen Konfigurationsdatei 5 abgelegt werden.

In einem Schritt 107 werden die Funktionseinheiten 8 des Computerprogramms 7 aktualisiert. Dies kann beispielsweise

30 durch den automatischen Aufruf eines Compilers 6 geschehen, der die in einem Source-Code vorliegenden Funktionseinheiten 8 in einen Maschinencode übersetzt. Dazu liest der Compiler 6 die implementierungsabhängigen Konfigurationsdateien 5 ein und steuert die Erzeugung des

35 Maschinencodes in Abhängigkeit der in den

implementierungsabhängigen Konfigurationsdateien 5  
abgelegten konkreten Konfigurationsdaten. Es ist auch  
vorstellbar, dass eine oder mehrere Funktionseinheiten 8  
bereits in Maschinencode vorliegen. In diesem Fall kann der  
5 Compiler beispielsweise den von den Scripten 4 erzeugten  
Source-Code (file\_2c, file\_kc in Figur 1) unter  
Berücksichtigung der Header-Dateien (file\_1.h, file\_2.h,  
file\_3.h) in Maschinencode übersetzen und den so  
übersetzten Maschinencode mittels eines dem Compiler 6  
10 zugeordneten sogenannten Linkers mit dem die  
Funktionseinheiten 8 repräsentierenden Maschinencode  
verbinden.

In einem Schritt 108 endet das Verfahren. In diesem Schritt  
15 ist das Computerprogramm 7 derart konfiguriert, dass die in  
den implementierungsunabhängigen Konfigurationsdateien  
abgelegten konkreten Konfigurationsdaten in dem erzeugten  
Maschinencode berücksichtigt werden.

20 Selbstverständlich ist es möglich, dass das Script 2  
und/oder die Scripte 4 in einer anderen Scriptsprache  
geschrieben sind oder als ausführbare Programme  
ausgestaltet sind.

25 Die in Figur 2 dargestellten Ausführungsschritte können  
selbstverständlich variieren und die Reihenfolge der  
Abarbeitung kann teilweise verändert werden. So ist es  
vorstellbar, dass die Plausibilitätsprüfung 104 durch eines  
oder mehrere der Scripte 4 durchgeführt wird.

30 Das Verfahren kann insbesondere auch von einer oder  
mehreren implementierungsunabhängigen Konfigurationsdateien  
ausgehen, ein oder mehrere Scripte 2 aufweisen, die  
beispielsweise hintereinander ausgeführt werden, ein oder  
35 mehrere Scripte 4 aufweisen, die jeweils ein oder mehrere

implementierungsabhängige Konfigurationsdateien 5 erzeugen und selbstverständlich kann das Computerprogramm 7 ein oder mehrere Funktionseinheiten 8 aufweisen. Mit dem erfindungsgemäßen Verfahren ist es insbesondere möglich zu  
5 erkennen, ob eine oder mehrere der Funktionseinheiten 8 bei der durch die implementierungsunabhängigen Konfigurationsdateien vorgegebenen Konfiguration tatsächlich zur Anwendung kommen. Ist dies nicht der Fall, kann dies durch ein dem Konfigurationsdatencontainer 3  
10 zugeordnetes, nicht dargestelltes Softwaretool erkannt werden. Dies ermöglicht es, dass eine derartige Funktionseinheit 8 nicht konfiguriert wird und mittels der implementierungsabhängigen Konfigurationsdateien 5 der Compiler 6 veranlasst wird, die Funktionseinheit 8 nicht  
15 mit in den zu erzeugenden Maschinencode zu übernehmen. Dadurch kann das erfindungsgemäße Verfahren besonders schnell durchgeführt werden. Der von einem Computerprogramm, das mittels des erfindungsgemäßen Verfahrens konfiguriert wurde, erzeugte Maschinencode kann  
20 dabei besonders kompakt sein und somit Speicherplatz einsparen.

Es ist möglich, dass das Script 2 selbst bereits die Erzeugung einer oder einer Mehrzahl von  
25 implementierungsabhängigen Konfigurationsdateien 5 veranlasst. Dadurch kann das erfindungsgemäße Verfahren besonders schnell durchgeführt werden. Dies kann beispielsweise vorteilhaft für abstrakte Konfigurationsdaten sein, die keine Abhängigkeiten  
30 aufweisen und sich von den konkreten Konfigurationsdaten unterscheiden.

5

## 10 Ansprüche

1. Verfahren zur Konfiguration mindestens eine Funktionseinheit umfassenden Computerprogramms, **gekennzeichnet durch** die folgenden Schritte:

- 15           - Erstellen mindestens einer implementierungsunabhängigen Konfigurationsdatei (1) und/oder Änderung von in der mindestens einen implementierungsunabhängigen Konfigurationsdatei (1) abgelegten Informationen;
- 20           - Automatischer Aufbau und/oder automatische Aktualisierung von in einem Konfigurationsdatencontainer (3) abgelegten Konfigurationsdaten in Abhängigkeit von den in der mindestens einen implementierungsunabhängigen Konfigurationsdatei (1) abgelegten Informationen;
- 25           - Automatisches Erzeugen mindestens einer implementierungsabhängigen Konfigurationsdatei (5) in Abhängigkeit von den in dem Konfigurationsdatencontainer (3) abgespeicherten Konfigurationsdaten;
- 30           - Automatische Konfiguration der mindestens einen Funktionseinheit in Abhängigkeit von in der



mindestens einen implementierungsabhängigen  
Konfigurationsdatei abgelegten Informationen.

2. Verfahren nach Anspruch 1 dadurch gekennzeichnet, dass  
mindestens eine Abhängigkeitsinformation automatisch  
5 erzeugt wird, die eine Abhängigkeit von mindestens zwei in  
dem Konfigurationsdatencontainer vorliegenden  
Konfigurationsdaten beschreibt, und dass die mindestens  
eine implementierungsabhängige Konfigurationsdatei in  
Abhängigkeit der mindestens einen Abhängigkeitsinformation  
10 erzeugt wird.
3. Verfahren nach einem der vorangehenden Ansprüche,  
wobei das Computerprogramm mehrere Funktionseinheiten  
aufweist, dadurch gekennzeichnet, dass mehrere  
implementierungsunabhängige Konfigurationsdateien erstellt  
15 werden und jede der implementierungsunabhängigen  
Konfigurationsdateien mindestens einer Funktionseinheit  
zugeordnet wird.
4. Verfahren nach einem der vorangehenden Ansprüche,  
wobei das Computerprogramm mehrere Funktionseinheiten  
20 aufweist, dadurch gekennzeichnet, dass mehrere  
implementierungsabhängige Konfigurationsdateien erzeugt  
werden und jede der implementierungsabhängigen  
Konfigurationsdateien mindestens einer Funktionseinheit  
zugeordnet wird.
- 25 5. Verfahren nach einem der vorangehenden Ansprüche,  
dadurch gekennzeichnet, dass die mindestens eine  
implementierungsabhängige Konfigurationsdatei in  
Abhängigkeit von mindestens einer Eigenschaft einer  
Hardware, auf der eine Installation zumindest eines Teils  
30 des konfigurierten Computerprogramms ermöglicht werden  
soll, erzeugt wird.

6. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass die mindestens eine implementierungsabhängige Konfigurationsdatei in Abhängigkeit von dem Ergebnis einer Plausibilitätsprüfung erzeugt wird.
7. Verfahren nach Anspruch 5 und 6, dadurch gekennzeichnet, dass zur Durchführung der Plausibilitätsprüfung die mindestens eine Hardware-Eigenschaft verwendet wird.
8. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass automatisch eine Dokumentation erstellt wird und die Dokumentation die innerhalb der mindestens einen implementierungsunabhängigen Konfigurationsdatei und/oder der mindestens einen implementierungsabhängigen Konfigurationsdatei abgelegten Informationen beschreibt.
9. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass die mindestens eine .. implementierungsunabhängige Konfigurationsdatei in einem XML-basierten Format erstellt wird.
10. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass in Abhängigkeit von den Konfigurationsdaten automatisch ermittelt wird, ob eine von dem Computerprogramm umfasste Funktionseinheit von dem Computerprogramm benötigt wird und eine Konfiguration dieser Funktionseinheit nur durchgeführt wird, falls die Funktionseinheit von dem Computerprogramm benötigt wird.
11. Softwaresystem zur Konfiguration eines mindestens eine Funktionseinheit umfassenden Computerprogramms, **dadurch gekennzeichnet**, dass das Softwaresystem aufweist:

- mindestens eine implementierungsunabhängige Konfigurationsdatei;
- einen Konfigurationsdaten umfassenden Konfigurationsdatencontainer und/oder Mittel zum Erstellen eines Konfigurationsdatencontainers in Abhängigkeit von in der mindestens einen implementierungsunabhängigen Konfigurationsdatei abgelegten Informationen;
- Mittel zum Ändern und/oder Auslesen von Konfigurationsdaten aus dem Konfigurationsdatencontainer;
- Mittel zum automatischen Erzeugen mindestens einer implementierungsabhängigen Konfigurationsdatei in Abhängigkeit von in dem Konfigurationsdatencontainer abgespeicherten Konfigurationsdaten; und
- Mittel zum automatischen Konfigurieren der mindestens einen Funktionseinheit in Abhängigkeit von in der implementierungsabhängigen Konfigurationsdatei abgelegten Informationen.

12. Softwaresystem nach Anspruch 11, dadurch gekennzeichnet, dass das Softwaresystem Mittel zur Durchführung eines Verfahrens nach einem der Ansprüche 2 bis 11 aufweist.

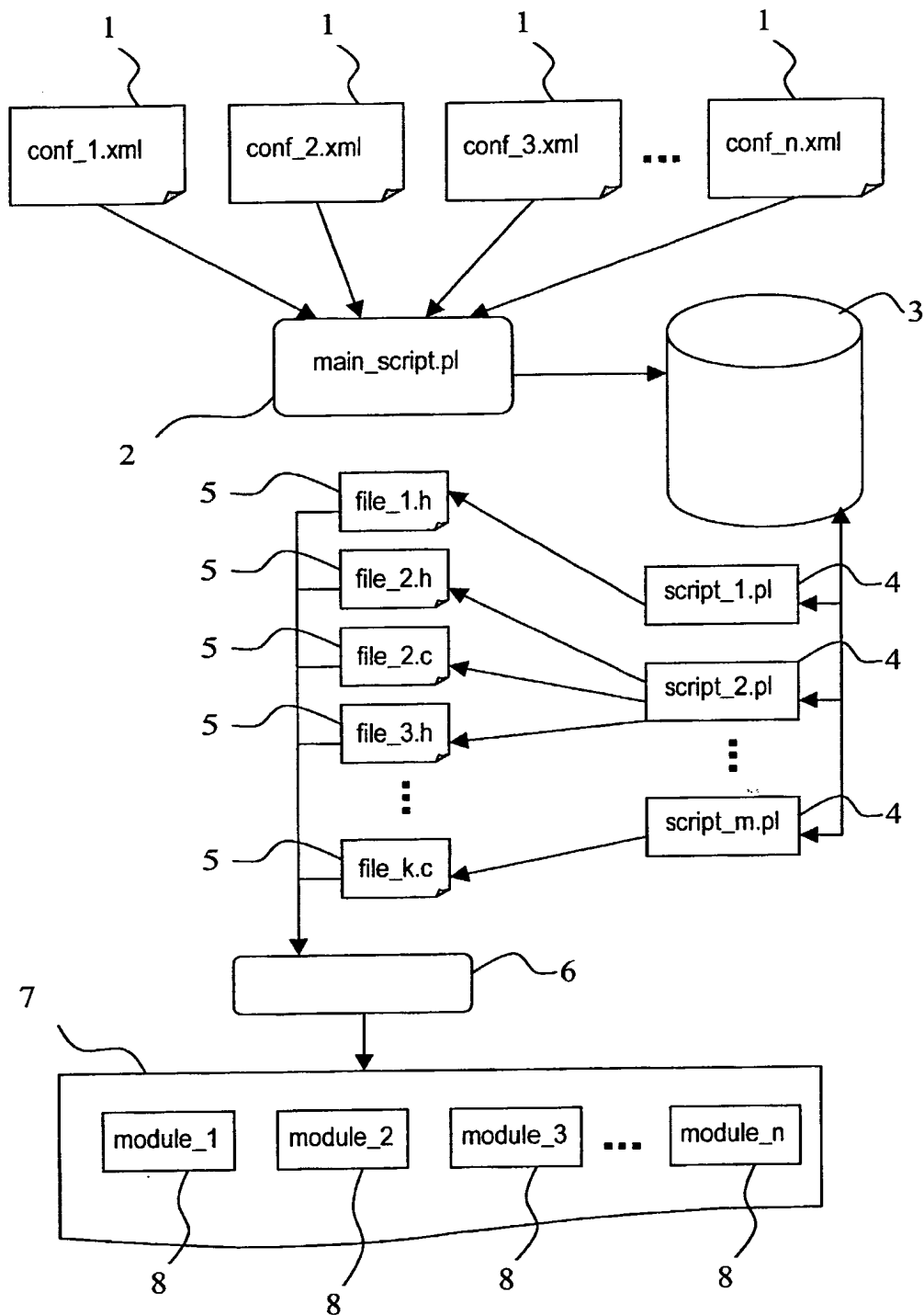
13. Softwaresystem nach Anspruch 11 oder 12, dadurch gekennzeichnet, dass das Softwaresystem auf einem Speichermedium abgespeichert wird.

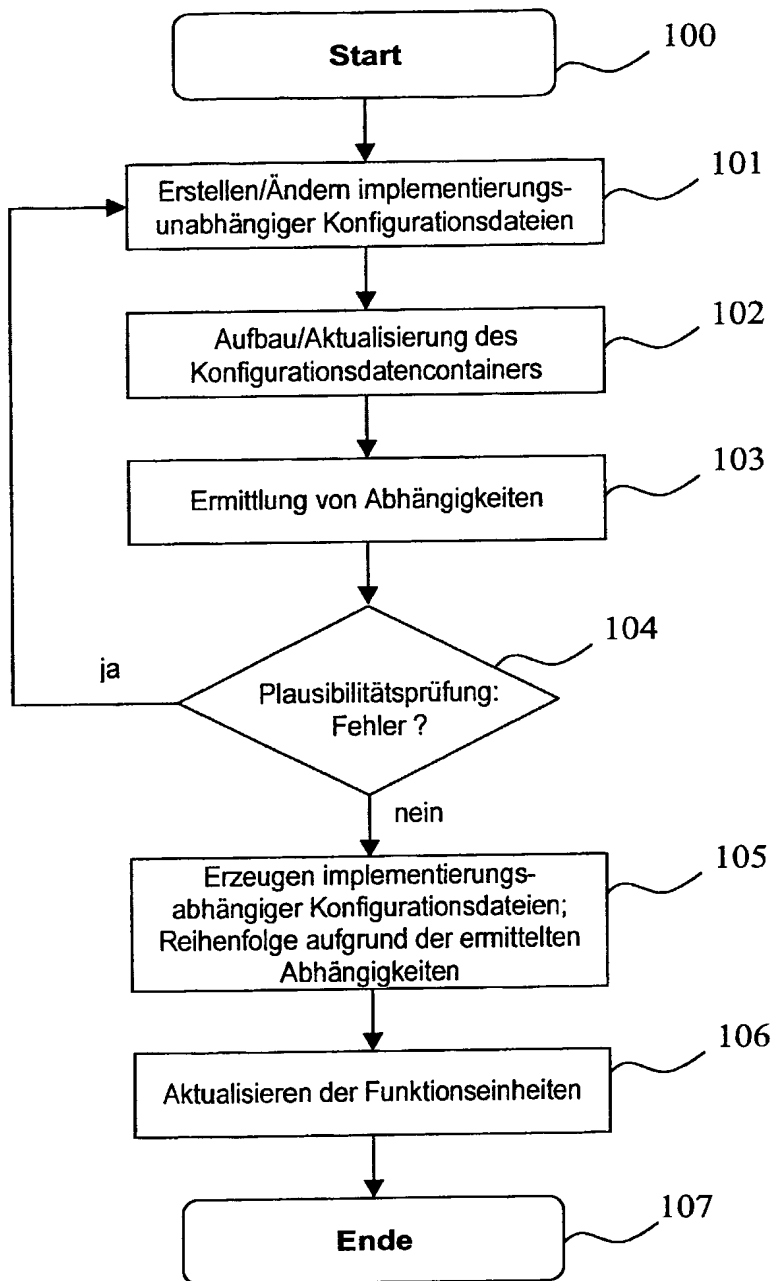
14. Softwaresystem nach Anspruch 13, dadurch gekennzeichnet, dass das Softwaresystem auf einem Random-

Access-Memory, auf einem Read-Only-Memory oder auf einem Flash-Memory abgespeichert ist.

15. Softwaresystem nach Anspruch 13, dadurch gekennzeichnet, dass das Softwaresystem auf einer Digital  
5 Versatile Disk (DVD), einer Compact Disc (CD) oder auf eine Festplatte (Hard Disc) abgespeichert ist.

16. Rechengerät, insbesondere Steuergerät, das einen Mikroprozessor aufweist, **dadurch gekennzeichnet**, dass das  
Rechengerät zur Durchführung eines Verfahrens nach einem  
10 der Ansprüche 1 bis 10 programmiert ist.

**Fig. 1**

**Fig. 2**

# INTERNATIONAL SEARCH REPORT

Int'l Application No  
PCT/EP2005/050289

**A. CLASSIFICATION OF SUBJECT MATTER**  
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)  
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 872 977 A (THOMPSON ET AL) 16 February 1999 (1999-02-16)	1,5,6, 10,11, 13-16
Y	column 3, line 1 - line 11  column 4, line 19 - line 43 column 7, line 61 - line 65	2-4,7-9, 12
Y	US 2002/040469 A1 (PRAMBERGER JOHANN) 4 April 2002 (2002-04-04) paragraph '0013! paragraph '0016! paragraph '0033! paragraph '0036! paragraph '0044! paragraph '0112! paragraph '0136! paragraph '0142!  ----- --/--	2-4,7-9, 12

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

\* Special categories of cited documents :

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*G\* document member of the same patent family

Date of the actual completion of the international search

11 May 2005

Date of mailing of the international search report

23/05/2005

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Ebert, W

## INTERNATIONAL SEARCH REPORT

International Application No  
PCT/EP2005/050289

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2002/199170 A1 (JAMESON KEVIN WADE) 26 December 2002 (2002-12-26) paragraph '0035! paragraph '0049! paragraph '0050! paragraph '0293! -----	1-16



# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/EP2005/050289

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 5872977	A	16-02-1999	NONE	
US 2002040469	A1	04-04-2002	DE 10121790 A1	13-12-2001
US 2002199170	A1	26-12-2002	CA 2352575 A1	21-12-2002

# INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen

PCT/EP2005/050289

## A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES

IPK 7 G06F9/44

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

## B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)

IPK 7 G06F

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)

EPO-Internal, WPI Data, INSPEC

## C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 5 872 977 A (THOMPSON ET AL) 16. Februar 1999 (1999-02-16)	1,5,6, 10,11, 13-16
Y	Spalte 3, Zeile 1 - Zeile 11  Spalte 4, Zeile 19 - Zeile 43 Spalte 7, Zeile 61 - Zeile 65	2-4,7-9, 12
Y	US 2002/040469 A1 (PRAMBERGER JOHANN) 4. April 2002 (2002-04-04) Absatz '0013! Absatz '0016! Absatz '0033! Absatz '0036! Absatz '0044! Absatz '0112! Absatz '0136! Absatz '0142!	2-4,7-9, 12

☒ Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

☒ Siehe Anhang Patentfamilie

\* Besondere Kategorien von angegebenen Veröffentlichungen :

\*A\* Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

\*E\* älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

\*L\* Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

\*O\* Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

\*P\* Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

\*T\* Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

\*X\* Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

\*Y\* Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

\*G\* Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche

11. Mai 2005

Absenddatum des internationalen Recherchenberichts

23/05/2005

Name und Postanschrift der Internationalen Recherchenbehörde

Europäisches Patentamt, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Bevollmächtigter Bediensteter

Ebert, W

## C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
A	US 2002/199170 A1 (JAMESON KEVIN WADE) 26. Dezember 2002 (2002-12-26) Absatz '0035! Absatz '0049! Absatz '0050! Absatz '0293! -----	1-16

# INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen

PCT/EP2005/050289

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung	Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
US 5872977	A	16-02-1999	KEINE		
US 2002040469	A1	04-04-2002	DE	10121790 A1	13-12-2001
US 2002199170	A1	26-12-2002	CA	2352575 A1	21-12-2002